Dokumentace úlohy SYN: Zvýraznění syntaxe v Pythonu 3 do IPP 2016/2017
Jméno a příjmení: Radek Mojžíš
Login: xmojzi07

1 Návrh

Nejprve proběhla dekompozice úlohy na podúlohy, čímž se konečné řešení značně zjednodušilo. Nejprve je potřeba načíst vstupní argumenty, následně zkontrolovat dostupnost vstupních a výstupních souborů, dále převést vstup na vnitřní interpretaci a nakonec samotné generování výstupu. Vzhledem k charakteru projektu nebylo zapotřebí implementovat vlastní třídy. Jejich použití by pravděpodobně pouze zbytečně znepřehlednilo a zesložitilo kód.

2 Implementace

Následuje stručný popis implementace jednotlivých částí projektu. Pro podrobnější informace můžete nahlédnout do samotného kódu.

2.1 Načítání vstupních parametrů

Načítání vstupních parametrů probíhá ve funkci `geto_ptions()` za použití funkce `getopts`. Funkce otevře vstupní soubory, popřípadě ponechá `STDIN` a nastaví flagy pro volitelná nastavení.

2.2 Načtení a konverze vstupu

Načtení vstupu probíhá ve funkcích `parse_fmt_file()` a `format_text()`. V `parse_fmt_file()` se načte řádek po řádku formátovací soubor (prázdné řádky jsou ignorovány) a rozdělí se na dvojice [regulární výraz, [seznam formátovacích příkazů]]. Následně se provede převod regulárních výrazů specifikovaných projektem na regulární výrazy, kterým rozumí python. To je zajištěno funkcí `normalise_regex()`. Dále se také převádí formátovací řetězce na THML tagy které se objeví ve výstupu, což se provádí ve funkci `normalise_format()`.

2.3 Syntaktická kontrola

Syntaktická kontrola probíhá částečně v `normalise_regex()` kde se detekují nepovolené kombinace symbolů. Dále proběhne detekce neplatných barev a velikostí. Pokud se narazí na neznámý formátovací řetězec program je ukončen s chybou.

2.4 Generování HTML elementů

Nejpre se vytvoří seznam dvojic `[[match_span], [formats]]`, kde `match_span` je pozice výsledku vyhledávání regulárního výrazu ve vstupním textu ve tvaru `[začátek, konec]` a `formats` je seznam formátovacích řetězců. Potom se vygeneruje finální seznam všech tagů podle `match_span`. Finální seznam je seznam dvojic `[index, tag]`, kde `index` je index, na který se má `tag` umístit. Tento seznam je řazen podle prvku `[index]`, kde levé tagy jsou při kolizi řazeny stabilně (pozdější vstup bude zařazen za prvky na stejném indexu) a právé tagy jsou řazeny opačně oproti stabilnímu řazení (pozdější vstup je zařazen před všechny prvky na stejném indexu). Toho je docíleno pomocným seznamem a použitím modulu `bisect`. Nakonec se všechny tagy postupně vloží na patřičnou pozici do výstupního řetězce.

2.5 Výstup

Před samotným zapsáním na výstup se případně před každý symbol `'\n'` přidá `<br />`, podle stavu proměnné `brflag`, která je nastavena na *True* v případě, že je zvoleno nastavení `--br`.

3 Závěr

Projekt nebyl nijak zásadně obtížný. Největší problém při implementaci byla možnost překrývání se tagů, kdy jeden tag může začít uvnitř dvojice tagů a končit až za ní. Tato funkcionalita samotná byla na implementaci asi nejsložitější z celého projektu.